Amendments to the Claims:

1. (Currently Amended) A method for supporting virtual machines in a data processing system, the method comprising:

executing an emulation patch for a guest virtual machine (VM) of a processing system, the emulation patch including data to facilitate identification of a routine for emulating a guest instruction;

in response to execution of the emulation patch, transferring control from the guest VM to a virtual machine monitor (VMM) without saving contextual data that defines a system state for the guest VM ~~a trap frame~~; and

using the data from the emulation patch to find an emulation routine for the guest instruction.

2. (Original) A method according to claim 1, wherein the operation of executing an emulation patch comprises executing an instruction that includes an immediate value to be used for finding the emulation routine.

3. (Original) A method according to claim 1, wherein the operation of executing an emulation patch comprises executing a flow control instruction, wherein the flow control instruction includes an address to be used for finding the emulation routine, the flow control instruction selected from a group consisting of:

a call instruction;

a jump instruction; and

a branch instruction.

4. (Original) A method according to claim 1, wherein the operation of executing an emulation patch comprises executing an instruction selected from the group consisting of:

a break instruction;

a branch instruction;

a call instruction; and

a jump instruction.

5. (Original) A method according to claim 1, further comprising:

determining an index, based at least in part on data produced by the emulation patch; and

using the index to find the emulation routine to be executed.

6. (Original) A method according to claim 1, further comprising:

automatically determining whether the guest instruction is to be patched for emulation, based at least in part on a list of instructions to be patched; and

inserting the emulation patch in response to a determination that the guest instruction is to be patched.

7. (Original) A method according to claim 1, further comprising:

automatically determining whether the guest instruction is to be patched for emulation, based at least in part on a list of instructions to be patched; and

retrieving a code template that corresponds to the guest instruction to be patched.

8. (Original) A method according to claim 1, further comprising:

automatically determining whether the guest instruction is to be patched for emulation, based at least in part on a list of instructions to be patched;

retrieving a code template that corresponds to the guest instruction to be patched; and

generating the emulation routine for emulating the guest instruction, based at least in part on the code template.

9. (Original) A method according to claim 1, further comprising:

automatically determining whether the guest instruction is to be patched for emulation, based at least in part on a list of instructions to be patched, wherein the guest instruction resides in a slot of an instruction bundle;

retrieving a code template that corresponds to the guest instruction to be patched; and

generating the emulation routine for emulating the guest instruction, based at least in part on the code template and on the slot containing the guest instruction.

10. (Currently Amended) A method according to claim 1, further comprising:

in response to execution of the emulation patch, ~~find~~ finding and executing the emulation routine for the guest instruction without decoding the guest instruction.

11. (Currently Amended) A processing system to support virtual machines, the processing system comprising:

a processor;

a machine-accessible medium responsive to the processor; and

instructions in the machine accessible medium, wherein the instructions, when executed by the processing system, cause the processing system to perform operations comprising:

executing an emulation patch for a guest virtual machine (VM) of the processing system, the emulation patch including data to facilitate identification of a routine for emulating a guest instruction;

in response to execution of the emulation patch, transferring control from the guest VM to a virtual machine monitor (VMM) without saving contextual data that defines a system state for the guest VM ~~a trap frame~~; and

using the data from the emulation patch to find an emulation routine for the guest instruction.

12. (Original) A processing system according to claim 11, wherein the emulation patch comprises an instruction with an immediate value, the immediate value to be used for finding the emulation routine.

13. (Original) A processing system according to claim 11, wherein the emulation patch comprises a flow control instruction with an address to be used for finding the emulation routine, the flow control instruction selected from a group consisting of:

a call instruction;

a jump instruction; and

a branch instruction.

14. (Original) A processing system according to claim 11, wherein the emulation patch comprises an instruction selected from the group consisting of:

a break instruction;

a branch instruction;

a call instruction; and

a jump instruction.

15. (Original) A processing system according to claim 11, wherein the instructions perform operations comprising:

determining an index, based at least in part on data produced by the emulation patch; and

using the index to find the emulation routine to be executed.

16. (Original) A processing system according to claim 11, wherein the instructions perform operations comprising:

automatically determining whether the guest instruction is to be patched, based at least in part on a list of instructions to be patched; and

inserting the emulation patch in response to a determination that the guest instruction is to be patched.

17. (Original) A processing system according to claim 11, wherein the instructions perform operations comprising:

automatically determining whether the guest instruction is to be patched, based at least in part on a list of instructions to be patched; and

retrieving a code template that corresponds to the guest instruction to be patched.

18. (Original) A processing system according to claim 11, wherein the instructions perform operations comprising:

automatically determining whether the guest instruction is to be patched, based at least in part on a list of instructions to be patched;

retrieving a code template that corresponds to the guest instruction to be emulated; and

generating the emulation routine for emulating the guest instruction, based at least in part on the code template.

19. (Original) A processing system according to claim 11, wherein the instructions cause the processing system to perform operations comprising:

in response to execution of the emulation patch, finding and executing the emulation routine for the guest instruction without decoding the guest instruction.

20. (Currently Amended) An apparatus to support virtual machines, the apparatus comprising:

a tangible machine accessible medium; and

instructions in the tangible machine accessible medium, wherein the instructions, when executed by a processing system, cause the processing system to perform operations comprising:

executing an emulation patch for a guest virtual machine (VM) of the processing system, the emulation patch including data to facilitate identification of a routine for emulating a guest instruction;

in response to execution of the emulation patch, transferring control from the guest VM to a virtual machine monitor (VMM) without saving contextual data that defines a system state for the guest VM ~~a trap frame~~; and

using the data to find an emulation routine for the guest instruction.

21. (Original) An apparatus according to claim 20, wherein the emulation patch comprises an instruction with an immediate value, the immediate value to be used for finding the emulation routine.

22. (Original) An apparatus according to claim 20, wherein the emulation patch comprises a flow control instruction with an address to be used for finding the emulation routine, the flow control instruction selected from a group consisting of:

a call instruction;

a jump instruction; and

a branch instruction.

23. (Original) An apparatus according to claim 20, wherein the emulation patch comprises an instruction selected from the group consisting of:

a break instruction;

a branch instruction;

a call instruction;

a jump instruction.

24. (Original) An apparatus according to claim 20, wherein the instructions perform operations comprising:

determining an index, based at least in part on data produced by the emulation patch; and

using the index to find the emulation routine to be executed.

25. (Original) An apparatus according to claim 20, wherein the instructions perform operations comprising:

automatically determining whether the guest instruction is to be patched, based at least in part on a list of instructions to be patched; and

inserting the emulation patch in response to a determination that the guest instruction is to be patched.

26. (Original) An apparatus according to claim 20, wherein the instructions perform operations comprising:

automatically determining whether the guest instruction is to be patched, based at least in part on a list of instructions to be patched;

retrieving a code template that corresponds to the guest instruction to be patched; and

generating the emulation routine for emulating the guest instruction, based at least in part on the code template.

27. (Original) An apparatus according to claim 20, wherein the instructions, when executed, cause the processing system to perform operations comprising:

in response to execution of the emulation patch, finding and executing the emulation routine for the guest instruction without decoding the guest instruction.